МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное
учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»
(ГУАП)

Составитель доц. к.т.н. Малаханов Р.Н.

ГЕНЕРАЦИЯ СИГНАЛОВ

Контрольная работа
для студентов заочной формы обучения по дисциплине
«Цифровые вычислительные устройства и микропроцессоры»

Санкт-Петербург

2020

Список сокращений

АС - аппаратный сброс

ДОП – демонстрационно-отладочная плата

МК – микроконтроллер

ПО – программное обеспечение

ПОП - подпрограмма обработки прерывания

ЦАП - цифро-аналоговый преобразователь

DAC – Digital-to-Analog Converter - ЦАП

Цель работы: изучение модулей таймера и ЦАП, встроенных в микроконтроллер 1986ВЕ93У, а также разработка программы, которая производит генерацию сигнала с заданными параметрами.

1. Методические указания

Описание модуля ЦАП приведено в спецификации МК [1] в разделе 24 на стр. с 328 по 330. В МК 1986ВЕ93У реализован только один ЦАП, который обозначен на рис. 93 как DAC2.

Описание модуля таймера приведено в [1] в разделе 22. Для выполнения контрольной работы следует ознакомиться с материалом, приведённым на стр. с 278 по 285 до подраздела 22.3.2.

Программное обеспечение

Для программирования ЦАП предназначены следующие функции.

1. Функция void DAC2_Inint(uint32_t DAC2_Ref) предназначена для настройки ЦАП. Входной параметр может принимать два значения:

- DAC2_AVCC – источником опорного напряжения является напряжение электрического питания с вывода AUcc (внутренние опорные напряжения 0 и 3,3 В);

- DAC2_REF - источником опорного напряжения является напряжение, подаваемое на выводы DAC2_REF микроконтроллера (внешнее опорное напряжение).

2. Функция void DAC2_Cmd(FunctionalState NewState) предназначена для включения или выключения модуля ЦАП. Вызов функции с входным параметром ENABLE включает ЦАП:

DAC2_Cmd(ENABLE).

После АС модуль ЦАП выключен.

3. Функция void DAC2_SetData(uint32_t Data) производит запись целого значения Data в ЦАП.

4. Функция

RST_CLK_PCLKcmd(RST_CLK_PCLK_DAC, ENABLE);

разрешает подачу сигнала тактовой частоты на ЦАП.

Для работы с таймером надо разрешить подачу на него сигнала тактовой частоты посредством:

- вызова функции

RST_CLK_PCLKcmd(RST_CLK_PCLK_TIMERx, ENABLE);

- установки разряда TIMxCLKEN в регистре TIM_CLOCK, формат которого приведён в [1] на стр. 177.

В обоих случаях вместо буквы x следует подставить порядковый номер таймера. Например, программный код для разрешения подачи сигнала тактовой частоты на таймер 1 будет выглядеть следующим образом:

RST_CLK_PCLKcmd(RST_CLK_PCLK_TIMER1, ENABLE);
MDR_RST_CLK->TIM_CLOCK = 0x01000000;,

где в значении 1000000h = 1 0000 0000 0000 0000 0000 0000_2 установлен 24 разряд TIM1CLKEN.

Для настройки таймера его вначале надо выключить посредством сброса нулевого разряда CNT_EN в регистре MDR_TIMERx->CNTRL. Формат регистра MDR_TIMERx->CNTRL приведён в [1] на стр. 301 и 302.

После окончания настройки таймера его надо включить посредством установки этого же разряда.

Как указано на стр. 280 в [1] для работы таймера надо задать значения TIMx_CNT, TIMx_PSG и TIM_ARR. Эти значения записываются соответственно в регистры таймера MDR_TIMERx->CNT, MDR_TIMERx->PSG и MDR_TIMERx->ARR, форматы которых приведены в [1] на стр. 300. При этом счётчик таймера будет работать на частоте

$$CLK = TIMx\_CLK/(TIMx\_PSG + 1),$$

где внутренняя частота TIMx_CLK = 8 МГц.

Время срабатывания таймера будет определяться выражением

$$T = TIMx\_ARR / CLK = TIMx\_ARR(TIMx\_PSG + 1)/TIMx\_CLK = \\ = TIMx\_ARR(TIMx\_PSG + 1)/8\cdot 10^6. \quad (1)$$

По умолчанию (после АС) сигналом для изменения TIMx_CNT является внутренняя частота TIMx_CLK; счётчик является суммирующим (счёт вверх). Именно эти параметры следует использовать в контрольной работе, поэтому дополнительная настройка таймера не требуется.

После истечения каждого интервала времени T будет вызываться ПОП

```
void Timerx_IRQHandler(void),
```

в тело которой следует вставить программный код для формирования сигнала. Также после срабатывания таймера будет установлен разряд CNTARREVENT в регистре MDR_TIMERx->STATUS, формат которого приведён в [1] на стр. 309 и 310. Этот разряд следует сбрасывать программно в теле ПОП.

Для разрешения генерации прерываний следует установить второй разряд CNTARREVENTIE в регистре MDR_TIMERx->IE, формат которого приведён в [1] на стр. 311 и 312, а также разрешить прерывания от таймера в контроллере прерываний посредством вызова функции

NVIC_EnableIRQ(Timerx_IRQn).

При вызове последней функции во входной константе Timerx_IRQn вместо буквы x следует подставить порядковый номер таймера.

Вызов функции RST_CLK_HSIadjust(23) устанавливает внутреннюю частоту TIMx_CLK равной 8 МГц.

2. Порядок выполнения контрольной работы

1. В соответствии со своим вариантом разработайте алгоритм работы программы для МК. Варианты индивидуальных заданий приведены в табл. 1. Номер варианта совпадает с последней цифрой в зачётной книжке; если последней цифрой является нуль, то следует выполнять 10 вариант. В табл. 1 приняты следующие обозначения:

- t_1 – длительность фронта импульса;
- t_2 – длительность среза импульса;
- τ – длительность импульса.

2. В соответствии с разработанным алгоритмом разработайте программу на языке программирования Си. ПО следует разрабатывать для ДОП 1986EvBrd_48, описание которой приведено в [2]. Электрическая принципиальная схема ДОП приведена в [3].

3. Составьте отчёт о выполнении контрольной работы.

В листинге 1 в качестве примера приведена программа, которая выполняет 11 вариант в табл. 1. На рис. 1 приведён сигнал, который генерирует программа.

Таблица 1 - Варианты заданий

№ варианта	Таймер	t_1, мс	t_2, мс	τ, мс	Период, мс	Ампли-туда, В	Вид сигнала
1	2	40	0	40	40	2,5	Треугольные импульсы
2	3	0	30	30	30	2	Треугольные импульсы
3	2	20	60	80	80	3,3	Треугольные импульсы
4	2	40	40	80	120	2,75	Треугольные импульсы
5	3	50	0	50	100	2,25	Треугольные импульсы
6	3	0	60	60	100	3,25	Треугольные импульсы
7	3	40	0	60	80	3,3	Трапецеидальные импульсы
8	2	0	20	50	90	2,5	Трапецеидальные импульсы
9	3	60	20	80	140	2,75	Треугольные импульсы
10	2	0	20	50	50	2,25	Трапецеидальные импульсы
11	1	30	10	60	110	3	Трапецеидальные импульсы

Листинг 1.

```
#include "MDR32F9Qx_config.h"
#include "MDR32Fx.h"
#include "MDR32F9Qx_timer.h"
#include "MDR32F9Qx_rst_clk.h"
#include "MDR32F9Qx_port.h"
#include "MDR32F9Qx_dac.h"

// Константы, задающие параметры сигнала
#define T1 240
#define T2 400
#define T3 480
#define T4 880
#define T5 80

void PortInit(void);

unsigned Timer1, yi;

int main(void)
{
```

```
    RST_CLK_PCLKcmd(RST_CLK_PCLK_RST_CLK, ENABLE);
    PortInit();

    // Timer 1
    RST_CLK_PCLKcmd(RST_CLK_PCLK_TIMER1, ENABLE);
    MDR_RST_CLK->TIM_CLOCK = 0x01000000;
    MDR_TIMER1->CNTRL = 0; // Выключение таймера
    MDR_TIMER1->CNT = 0; // Установка периода срабатывания таймера
    MDR_TIMER1->PSG = 0;
    MDR_TIMER1->ARR = 1000;
    MDR_TIMER1->IE = 2; // Установка разряда CNTARREVENTIE
    MDR_TIMER1->CNTRL = 1; // Включение таймера
    NVIC_EnableIRQ(Timer1_IRQn); // Разрешение прерываний

    // ЦАП
    RST_CLK_PCLKcmd(RST_CLK_PCLK_DAC, ENABLE);
    DAC2_Init(DAC2_AVCC); // Установка опорных напряжений 0 и 3,3 В
    DAC2_Cmd(ENABLE); // Включение АЦП

    RST_CLK_HSIadjust(23);

while(1)
{
    static unsigned t = 0;
    static float y;
    // k1 используется для формирования фронта, а k2 – для среза.
    static const float k = 4095./3.3, k1 = 3./(float)T1, k2 = 3./(float)T5;

    // Если прерывание от таймера произошло, то следует рассчитать новое значение yi
    // для последующей записи в ЦАП после следующего срабатывания таймера
    if(Timer1)
    {
        Timer1 = 0;
        // Фронт импульса
        if(++t <= T1) //  T1* ARR/8 МГЦ = 30 мс
        {
            y = k1*(float)t; // вычисление амплитуды сигнала в вещественной форме
            yi = (unsigned)(y * k); // получение целого значения для ЦАП, формула (2)
        }
        else if(t > T1 && t <= T2) // плоский участок длительностью 20 мс
            yi = 0xE8B; // Значение E8Bh соответствует напряжению 3 В
        else if(t > T2 && t < T3) // срез импульса
        {
            y = 3. - k2*(float)(t - T2);
            yi = (unsigned)(y * k);
        }
        else if(t >= T3 && t < T4) // напряжение равно нулю между 60 и 110 мс
            yi = 0;
        else
            t = 0; // начало формирования нового импульса
```

```
        }
    }
}

void PortInit(void)
{
    static PORT_InitTypeDef PortInit;

    RST_CLK_PCLKcmd( RST_CLK_PCLK_PORTE, ENABLE);

    // PORTE – выход ЦАП
    PortInit.PORT_Pin   = PORT_Pin_0;
    PortInit.PORT_OE    = PORT_OE_OUT;
    PORT_Init(MDR_PORTE, &PortInit);
}

void Timer1_IRQHandler(void)
{
    DAC2_SetData(yi); // запись в ЦАП рассчитанного значения
    Timer1 = 1; // Временной интервал закончился
    // Сброс флага CNT_ARR_EVENT
    MDR_TIMER1->STATUS &=~0x0002;
}
```

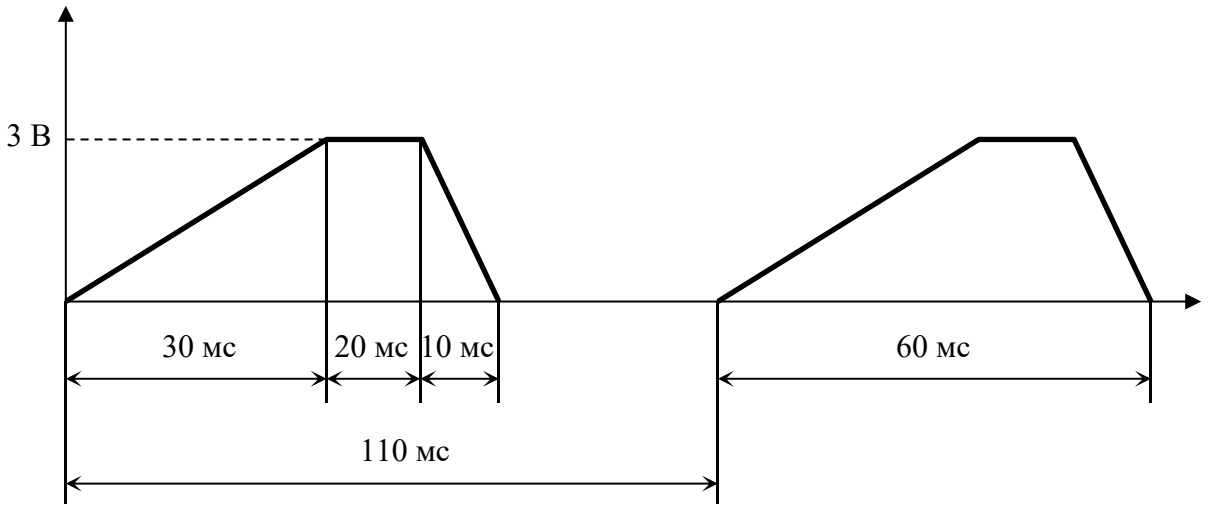

Рисунок 1 – Генерируемый сигнал. По оси абсцисс отложено время, а по оси ординат - напряжение.

Функция void PortInit(void) настраивает вывод PE0 порта E как аналоговый вывод, потому что выход ЦАП поступает на этот вывод.

При настройке таймера параметры TIM1_PSG и TIM1_ARR устанавливаются равными соответственно 0 и 10^3, поэтому в соответсвии с формулой (1) интервал срабатывания таймера 1

$$T = TIM1\_ARR//8\cdot10^6 = 1000/8\cdot10^6 = 125\cdot10^{-6} = 125 \text{ мкс.}$$

Таким образом, через каждые 125 мкс будет вызываться ПОП

void Timer1_IRQHandler().

В программе константа k = 4095/3,3 = 1240,91 является коэффициентом преобразования входного цифрового двоичного кода ЦАП в соответствующее ему выходное напряжение. ЦАП является 12-разрядным, поэтому входной цифровой двоичный код изменяется в диапазоне от 0 до 2^{12}-1=4095. Диапазон выходного напряжения ЦАП от 0 до 3,3 В, поэтому входной цифровой двоичный код n связан с выходным напряжением U соотношением

$$n = INT[kU], \quad (2)$$

где INT[] - функция, возвращающая целое значение. Например, если на выходе ЦАП надо установить напряжение, равное 3 В, то в ЦАП следует записать значение

$$n = INT[k\cdot U] = INT[1240,91\cdot3] = INT[3722,7] = 3723 = E8Bh.$$

В программе в качестве n используется переменная yi, а в качестве U - переменная y. Переменная y определяет текущее значение, которое должно быть на выходе ЦАП.

При вызове ПОП производятся следующие действия:

- запись предварительно рассчитанного значения yi в ЦАП;

- присвоение глобальной переменной Timer1 значения, равного единице;

- сброс разряда CNTARREVENT в регистре MDR_TIMER1->STATUS.

Переменная Timer1 используется в основной программе следующим образом: пока она равно нулю, никакие действия не совершаются. Если она равна единице, то это обозначает, что таймер сработал (закончился очередной дискретный интервал времени длительностью 125 мкс), текущее значение yi записано в ЦАП и следует рассчитывать следующее значение yi.

Переменная t предназначена для хранения текущего количества срабатываний таймера и инкрементируется после каждого срабатывания таймера.

Длительность фронта импульса сигнала равна 30 мс, что составляет

$$T1 = 30 \text{ мс} / 125 \text{ мкс} = 240$$

срабатываний таймера. Таким образом, фронт импульса сигнала формируется при значении переменной t в диапазоне от 0 до 240.

Длина плоского участка импульса сигнала равна 20 мс, что составляет 20 мс/125 мкс = 160 срабатываний таймера. Плоский участок начинается при значении t = T1 + 1 = 241 и оканчивается при t = T2 = T1 + 160 = 400. Аналогичным образом определяются константы длительности импульса T3 и длительности среза импульса T5. Константа T4 = 110 мс/125 мкс = 880 определяет период сигнала. После того, как переменная t достигает значения T4, ей присваивается значение 0 и начинается формирование нового импульса.

Константы k1 и k2 устанавливают скорости изменения соответственно фронта и среза импульса.

3. Оформление отчёта

Отчёт должен быть оформлен в соответствии с требованиями нормоконтроля и должен содержать:

- цель работы;
- блок-схему алгоритма программы;
- текст программы на языке программирования Си;
- результаты расчётов с приведением расчётных формул;
- библиографический список;
- выводы о проделанной работе.

Библиографический список

1. Микросхемы 32-разрядных однокристальных микро-ЭВМ с памятью Flash-типа 1986ВЕ9xy, К1986ВЕ9xy, К1986ВЕ92*QI*, К1986ВЕ92*QC*, 1986ВЕ91Н4, К1986ВЕ91Н4, 1986ВЕ94Н4, К1986ВЕ94Н4 [Электронный ресурс]: Спецификация. – Версия 3.20.1 от 15.05.2020. – Электрон. дан. (6,89 Mbytes). – [Б.м.]: АО «ПКК Миландр», 2020. – Режим доступа: https://ic.milandr.ru/upload/iblock/639/639018f8d49abe275ecb106e968df014.pdf, свободный. – Загл. с экрана.

2. Демонстрационно-отладочная плата 1986*EvBrd*_48 [Электронный ресурс] : Техническое описание. – Версия 1.0 от 25.05.2010. – Электрон. дан. (1,05 Mbytes). – [Б.м.]: ЗАО «ПКК Миландр», 2010. – 1 электрон. опт. диск (CD-ROM). – Систем. требования: Pentium 100МГц ; 16 Мб RAM ; Windows 7 ; CD-ROM дисковод ; SVGA видеокарта, 256 цв. ; мышь. - Загл. с экрана. - CD-ROM входит в комплект поставки демонстрационно-отладочной платы 1986*EvBrd*_48.

3. Отладочная плата 1986ВЕ93У [Электронный ресурс] : Схема электрическая принципиальная. – Revision 3. Последнее изменение

24.04.2014. – Электрон. граф. дан. (148 Kbytes). – [Б.м.]: ЗАО «ПКК Миландр», 2014. - 1 электрон. опт. диск (CD-ROM). – Систем. требования: Pentium 100МГц ; 16 Мб RAM ; Windows 7 ; CD-ROM дисковод ; SVGA видеокарта, 256 цв. ; мышь. – Загл. с экрана. - CD-ROM входит в комплект поставки демонстрационно-отладочной платы 1986*EvBrd*_48.